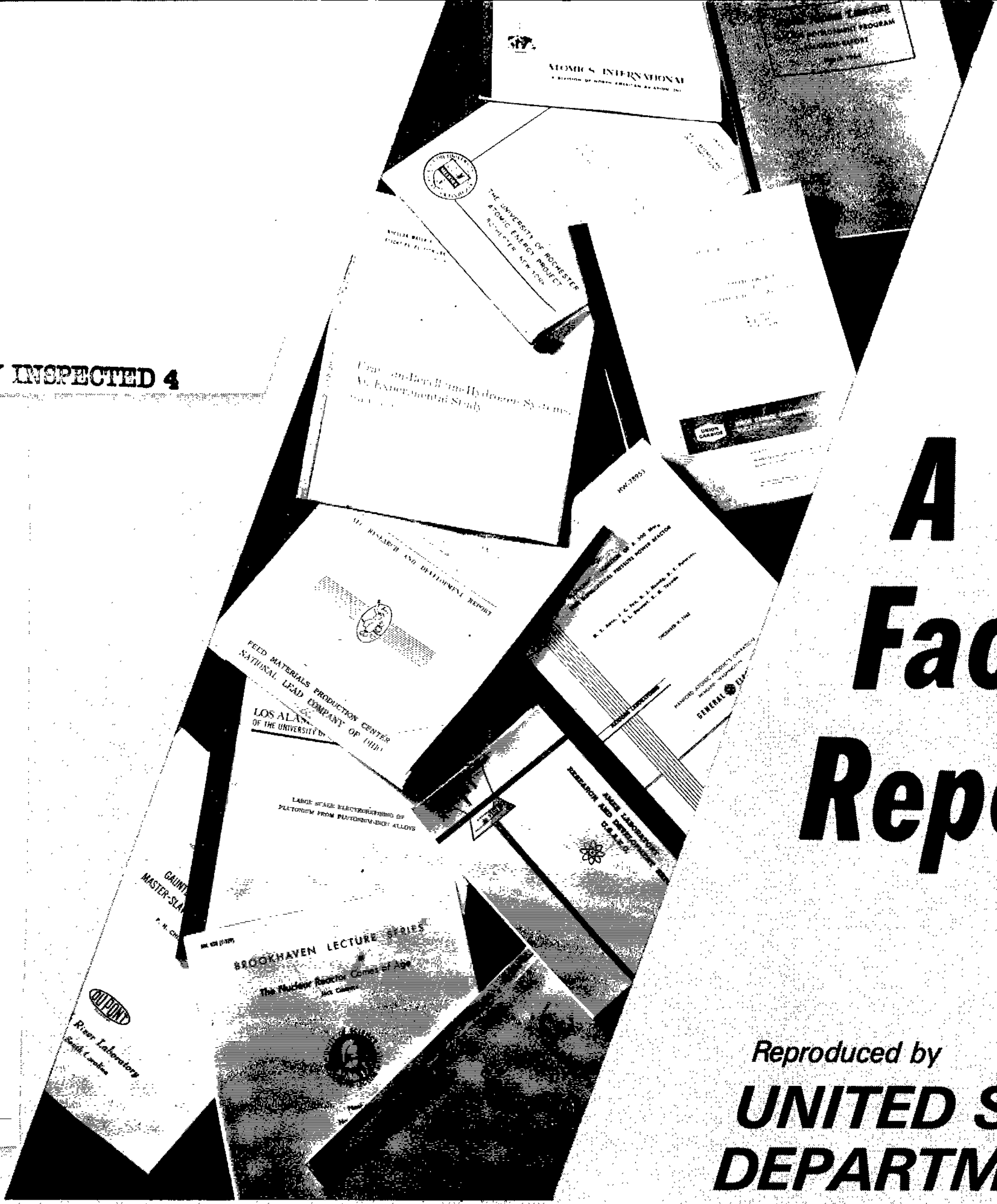
ATOMICS INTERNATIONAL
THE UNIVERSITY OF ROCHESTER ATOMIC ENERGY PROJECT ROCHESTER NEW YORK
FEED MATERIALS PRODUCTION CENTER NATIONAL LEAD COMPANY OF OHIO
LOS ALAMOS
BROOKHAVEN LECTURE SERIES
DU PONT

A Facsimile Report

Reproduced by

UNITED STATES DEPARTMENT OF ENERGY

Office of Scientific and Technical Information

Post Office Box 62 Oak Ridge, Tennessee 37831

U3357

Accession Number: 3357

Publication Date: Dec 01, 1991

Title: Massively Parallel Computing at Sandia and Its Application to National Defense (Preprint)

Personal Author: Dosanjh, S.S.

Corporate Author Or Publisher: Sandia National Laboratories, Albuquerque, NM 87185 Report Number: SAND-91-0278C

Report Prepared for: U.S. Dept. of Energy

Descriptors, Keywords: Parallel Processing Computer MIMD SRIM SDI Tracking Application SAR Model Simulation

Pages: 00010

Cataloged Date: Jan 30, 1992

Contract Number: DE-AC04-76DPOO789

Document Type: HC

Number of Copies In Library: 000001

Original Source Number: DE91008178

Record ID: 23171

Source of Document: DOE

CONF-911129--1

SAND--91-0278C

DE91 008178

MASSIVELY PARALLEL COMPUTING AT SANDIA AND ITS APPLICATION TO NATIONAL DEFENSE

SUDIP S. DOSANJH
Sandia National Laboratories
Albuquerque, New Mexico

ABSTRACT

Two years ago, researchers at Sandia National Laboratories showed that a massively parallel computer with 1024 processors could solve scientific problems more than 1000 times faster than a single processor. Since then, interest in massively parallel processing has increased dramatically. This review paper discusses some of the applications of this emerging technology to important problems at Sandia. Particular attention is given here to the impact of massively parallel systems on applications related to national defense. New concepts in heterogenous programming and load balancing for MIMD computers are drastically increasing synthetic aperture radar (SAR) and SDI modeling capabilities. Also, researchers are showing that the current generation of massively parallel MIMD and SIMD computers are highly competitive with a CRAY on hydrodynamic and structural mechanics codes that are optimized for vector processors.

INTRODUCTION

Two years ago, Sandia National Laboratories won the Gordon Bell Prize, the Karp Challenge and a R&D 100 award for its pioneering research in massively parallel computing.[1] A major contribution of this work was to show that speedups of greater than 1000 could be achieved on scientific problems using a first generation massively parallel machine with 1024 processors -- previously, conventional wisdom was that it would be difficult to use as many as one hundred processors to solve a single problem. Researchers at Sandia and elsewhere are now showing that the current (second) generation of massively parallel computers manufactured by nCUBE, Thinking Machines, INTEL, and others can outperform traditional vector supercomputers on a wide range of applications.[2-5] In the next few years, significant advances in massively parallel systems are anticipated as more processors are added and the computational power of each processor increases.

Conventional vector processors, on the other hand, are not expected to make major gains in the near future. During the past decade, clockspeeds on the fastest supercomputers have increased by a factor of two. While a factor of two might be a major improvement in many fields, it is very small in the computer industry. By contrast, the microprocessor revolution has led to order of magnitude improvements in desktop computers. The slow increase of supercomputer speeds is attributable to a fundamental physical barrier -- namely, the speed of light. Clockspeeds on vector supercomputers are now measured in nanoseconds. For reference, one nanosecond is approximately the time it takes light to travel a foot. Dramatic increases in performance will not occur without a major technological breakthrough in, for example, high temperature superconductivity, that allows the size of these supercomputers to be decreased dramatically.

It is becoming increasingly evident that massively parallel systems represent the future of supercomputing. All of the major supercomputer vendors, including CRAY, DEC and IBM, have recently announced major projects to develop massively parallel machines. The Japanese government is now investing heavily in this technology and on a recent scientific visit to the Soviet Union, Sandia researchers learned that the Soviets have developed a 128 node parallel computer that gives CRAY-class performance.

During the last two years, Sandia's massively parallel computing effort has grown to more than fifty researchers. Large-scale scientific calculations are being conducted on an nCUBE 2, a Connection Machine (CM-2), a first generation nCUBE/ten (sometimes referred to as an nCUBE 1), and a network of workstations. Sandia is also a member of a consortium headed by Caltech that is purchasing a large INTEL Delta prototype. The nCUBE 2 at Sandia has 1024 processors, 4 Gigabytes of memory, and will soon have a 16 Gigabyte parallel disk system. Typical applications are executed at 1 to 2 billion calculations per second (Gigaflops) and the speed can be increased to over 10 Gigaflops by going to a system with the maximum possible 8192 processors. The CM-2 at Sandia has 16384 processors, 2 Gigabytes of memory and a 10 Gigabyte disk farm. Currently, as much as half of the total usage of our nCUBE 2 and CM-2 is by external researchers at universities, other national laboratories and private companies.

These massively parallel computers are being used at Sandia to solve a wide range of problems involving, for example, radar imaging, SDI tracking and correlating, shock physics, structural mechanics, fluid mechanics, heat transfer, the motion of charged particle beams through accelerators, molecular dynamics and quantum chemistry. In order to support these activities, research is conducted in mathematical algorithms, load balancing, domain decomposition, grid generation, and graphics. The work in mathematical algorithms includes both the development of new methods (e.g., parallel time stepping, which is a technique to accelerate time marching on parallel computers) and the parallel implementation of existing techniques (e.g, multigrid and conjugate gradient based schemes for solving systems of linear equations arising from the solution of partial differential equations).[6-8] Novel dynamic load balancing techniques have been constructed for Monte Carlo problems and particle simulations -- similar techniques are being investigated for hydrodynamics and shock physics computer codes.[2]

Work in graphics is being stressed because the problems being solved on massively parallel computers are large. One Gigabyte of data is often considered small in the applications of interest. It is next to impossible to analyze and evaluate results from such calculations without visualization. All of the large Sandia parallel machines (the nCUBE 2, CM-2 and nCUBE/ten) have graphics systems and a Stardent GS2000 with a Application Visualization System (AVS) is available.

This review paper discusses some of the codes that are being implemented on massively parallel computers at Sandia. Particular attention is given to codes that are used extensively to support national defense programs. The following sections discuss the parallel implementation and performance of radar imaging, SDI tracking and correlating, shock physics and structural mechanics models.

SYNTHETIC APERATURE RADAR SIMULATIONS

Synthetic-aperture radar (SAR) is often used in reconnaissance and remote sensing applications. SAR systems transmit and receive coherent broadband signals that are processed and focused to produce terrain images. These systems can be used at day or night and in all types of weather.

Computer simulations are needed to assess the performance of SARs. These models use ray tracing techniques similar to those employed in computer graphics. In military applications, difficulties arise because objects of interest have many edges and multiple reflections can occur. Consequently, as many as a million rays are needed to provide adequate resolution. It can easily take more than an hour to construct a complicated image using a conventional supercomputer. This is due to low efficiencies resulting from a poor match between the ray tracing problem and vector architectures. Because thousands of orientations are often required, many problems of interest are intractable using CRAY-type computers.

A massively parallel, MIMD version of a radar imaging code, SRIM, has been developed at Sandia. The beam is modeled as a collection of rays that leave the radar and bounce off targets, and a radar or optical image is constructed from ray-target intersections. The geometry of the objects and their reflection characteristics are specified a priori in the model (see Fig. 1).

The output of SRIM is an idealized, noise-free radar image whose resolution is limited only by the number of rays used. A real radar image has noise and finite resolution, both of which can be modeled accurately. A new parallel pro-

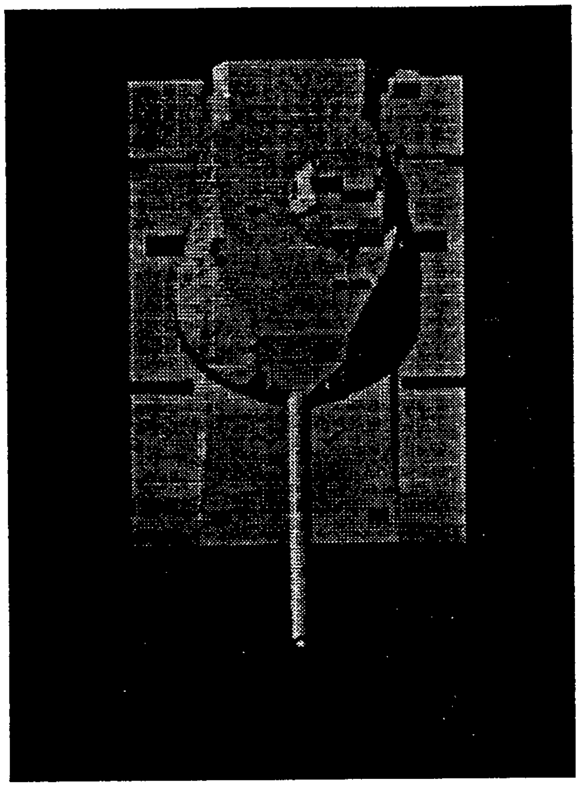

Fig. 1. Sample input for a SRIM calculation.

gram, ICON, has been developed to convert a SRIM ideal radar image to a realistic image that might actually be produced by a SAR system. The realistic image is constructed by adding random noise and convolving the data with a response function. The convolution is carried out using Fourier transforms on a hypercube. The parallel Fast Fourier transforms that have been developed for this work are very efficient (approximately three times as fast as comparable CRAY algorithms). These FFT routines can also be used for problems in fluid mechanics, quantum chemistry (e.g., electronic structure) and electromagnetism.

Extraordinary parallel performance has been achieved for SRIM by using new heterogeneous programming concepts and a dynamic load balancing algorithm. Typically, a 1024 processor first generation MIMD computer is about a factor of eight faster than a CRAY -- a second generation MIMD machine with 1024 processors is a factor of forty faster. Therefore, problems that take more than an hour on the CRAY are executed in minutes on massively parallel computers.

The parallel version of SRIM takes full advantage of the special characteristics of MIMD computers by executing several cooperating programs simultaneously in a heterogeneous manner. The serial version of SRIM consists of two codes; one traces rays and the other generates radar images from ray histories. These codes are executed consecutively. After calculating ray trajectories, results are written to a large intermediate file. This file is then used as input for the imaging calculation. In the MIMD version of SRIM, both of these programs are executed simultaneously by different sets of processors. Data is transferred from the ray tracing nodes to the image construction nodes in real-time through a very high bandwidth connection.

Typically, on a 1024 processor nCUBE 2 computer, 508 processors calculate beam paths and 508 construct partial images (each of these image construction nodes receives data from a single ray tracing processor). Of the remaining eight nodes, four are used to manage the calculation, by means of dynamic load balancing, while the other four help integrate the partial images into a complete image at the end of the calculation. The collected image is written to a disk to await further processing; it can also be sent to the real-time graphics system on the nCUBE 2 or the nCUBE/ten for immediate display.

Less than five-percent of the 30,000 lines of FORTRAN in SRIM were changed to develop the parallel version. This is partially due to the excellent match between ray tracing problems and MIMD architectures. Ray trajectories are independent and, for almost all problems of interest, the objects' geometry can be stored in every processor's memory.

In problems of this type, computational performance can degrade significantly due to load imbalances. This is due to the fact that different ray trajectory calculations can expend vastly varying amounts of computer time -- some rays undergo multiple reflections and eventually return to the radar while others miss the object completely and leave the domain of interest early in the computation. Because static task assignments can lead to poor parallel performance, dynamic methods have been investigated.

A unique Sandia contribution in this area has been the development of a dynamic load balancing algorithm that uses more than one node as a manager. Each manager is assigned a subset of the ray tracing processors and is given a list of tasks to accomplish. Currently, task assignments to these managers are static. The managers, in turn, give work to their processors dynamically. Whenever a worker processor completes its assigned tasks, it sends a message to its manager and receives additional work.

The optimum number of managers for a wide range of SAR simulations on a 1024 processor nCUBE/ten is eight or sixteen. There is an increase in the total execution time when the number of managers is increased or decreased significantly out of this range -- however, the minimum is fairly shallow so that small changes in the number of managers will not have a large impact on execution times. With fewer than four managers, many worker nodes are idle as they wait for tasks. The managers become overloaded as anywhere from a hundred thousand to a million messages are received. On the other hand, computational resources are wasted when more than thirty-two processors are used as managers. On a 1024 processor nCUBE 2, cut-through routing has a dramatic impact on the load balance process; in this case, four managers result in the fastest execution time.

Overall, the use of dynamic load balancing methods, instead of static task assignments, improves the execution time on a MIMD computer by a factor of two to three. Approximately thirty percent of this gain is due to using multiple managers.

Gains in speed resulting from the use of massively parallel computers and dynamic load balancing techniques are allowing researchers to construct image databases for objects in days instead of months, drastically increasing our capabilities in this area. It is also important to note that the load balancing algorithms developed here have broad applicability to Monte Carlo methods which are used, for example, to simulate electron microscope performance, investigate phase changes in materials, and study radiative heat transfer.

SDI TRACKING AND CORRELATING

Success of the Strategic Defense Initiative depends heavily on the development of computational tools to track the motion of a large number of projectiles. Because each missile can hold multiple warheads and decoys, the computer model must calculate the trajectories of more than one-hundred thousand objects. Furthermore, results are only useful if they can be provided in real-time. The input to the computer model is a series of sensor scans from various angles, provided as often as every two seconds. The goal is to convert these two-dimensional images into three-dimensional trajectories. The TRC code was developed to calculate trajectories of missiles during the midcourse phase (i.e., from the time they leave the atmosphere to when they re-enter). Once multiple warheads and decoys are released from a missile, they follow Kepler orbits.

A massively parallel version of the TRC code has been implemented at Sandia on a 1024 processor MIMD ma-

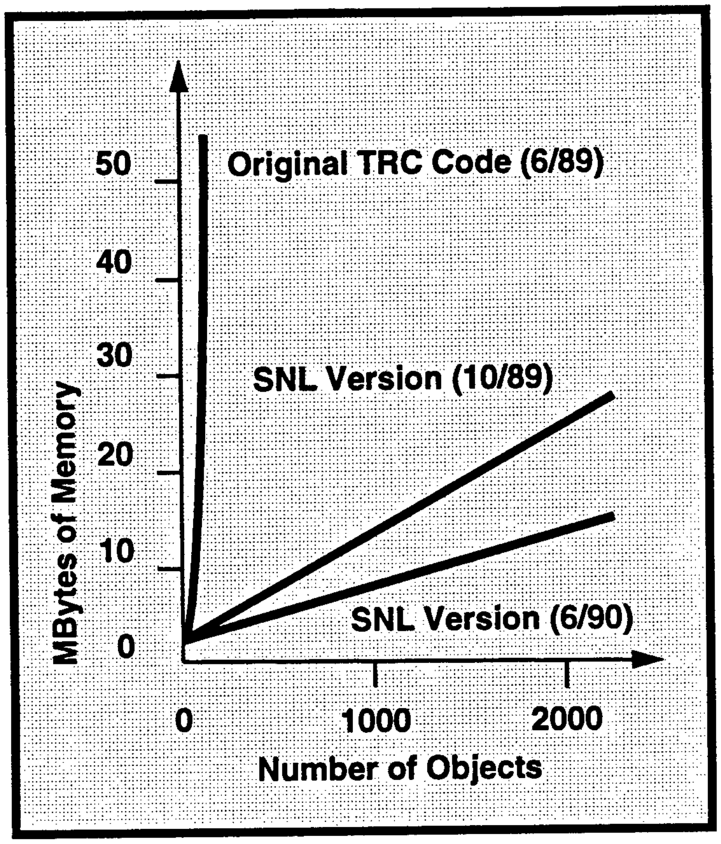

Fig. 2. Computer memory needed to solve tracking problems of various sizes.

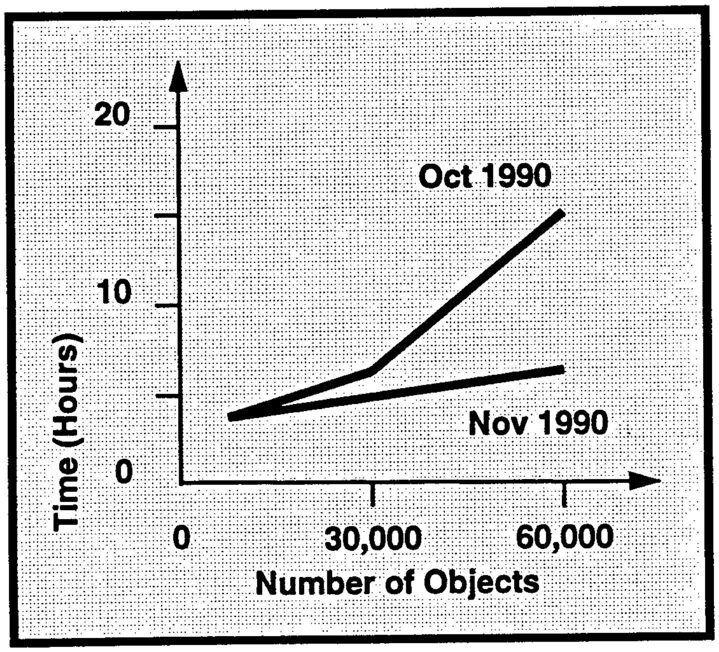

Fig. 3. The dependence of execution time on the number of objects has been reduced significantly. In the course of solving larger problems, Sandia researchers discovered and removed numerous bottlenecks in the algorithm. These bottlenecks have a tremendous impact on simulations with many objects, and almost no effect on small problems.

chine.[3] Most of the processors track objects -- however, a few manager (or gating) nodes are used to provide updated information as new sensor reports are generated. At the start of a calculation, several objects are assigned to each tracking processor. During the simulation, the gating nodes analyze new sensor reports and determine which objects belong to which processors. These managing nodes then send the appropriate updated positions to each tracking processor.

Two years ago, the largest tracking problem that had been solved on a traditional supercomputer involved 128 projectiles. Simulations with more than 50,000 objects are now routinely conducted on massively parallel computers. This is partially due to dramatic decreases in the amount of memory required to solve a given problem (see Fig.2). In the original code, the memory required increased quadratically with the number of projectiles. Sandia modifications have resulted in a linear dependence, drastically increasing the size of the problems that could be addressed.

Another important result is that the execution time of the massively parallel model increases only logarithmically with the number of projectiles. Therefore, as evidenced by Fig.3, only a limited amount of additional computer time is needed to investigate larger problems. The TRC code is currently executing three to four times faster on a 1024 processor nCUBE/ten than on a CRAY-YMP processor. This translates into a performance gain of twenty to thirty times over the CRAY on a nCUBE 2 with 1024 processors.

SHOCK PHYSICS

The ability to model shock physics is critical when studying high speed collisions between objects. Simulations often provide information that is not available from experiments and can help design and analyze experiments. The CTH code was developed at Sandia to investigate a wide spectrum of shock physics problems ranging from armor/antiarmor performance to weapons safety.[9] This code has recently been used to analyze potential damage to a NASA space station from collisions with debris. Natural debris, in the form of micrometeorites, as well as man-made debris from vehicles launched by the vast array of countries with space programs, are of concern.

It has become increasingly evident that an adequate three-dimensional shock physics modeling capability cannot be developed using current vector supercomputers. This is due to both speed and memory constraints. Two-dimensional calculations can take many hours on the fastest vector computers. Order of magnitude increases in computer speed are needed to solve three-dimensional problems in any reasonable time frame. Even if speed were not a consideration, the memory on these machines is not large enough to provide adequate grid resolution. Massively parallel computers will let researchers overcome these hurdles and develop a credible three-dimensional capability. In addition, these computers will help researchers solve two-dimensional problems faster and more economically.

An effort is underway to develop massively parallel versions of the CTH code for MIMD and SIMD architectures. CTH is a fully -compressible, multi-material shock hydrodynamics code. A high-resolution interface tracker is used to prevent distortion of material interfaces. High temperatures and multi-phase equations of state (solid, liquid and vapor) are considered in the model. Elastic-plastic behavior, high explosive detonations and fracture are also treated. An Eulerian mesh is used to solve the partial differential equations and higher-order differencing is employed on most approximations to the governing equations.

In the parallel implementation, each processor is assigned a portion of the computational grid. This process is often referred to as domain decomposition. In CTH, regular rectangular and cylindrical geometries are of particular interest. Processors are assigned subgrids with equal numbers of grid points and interprocessor communication is minimized by using subgrids that are rectangular or regular sections of a cylinder. Domain decomposition for a rectangular grid is illustrated in Fig. 4.

In a CTH calculation, interprocessor communication is very localized. Processors need information from a neighbor's subgrid during material interface reconstruction and when derivatives are differenced. However, data is only needed from the outer layer of a neighbor's subgrid. In order to facilitate the exchange of information between processors, a

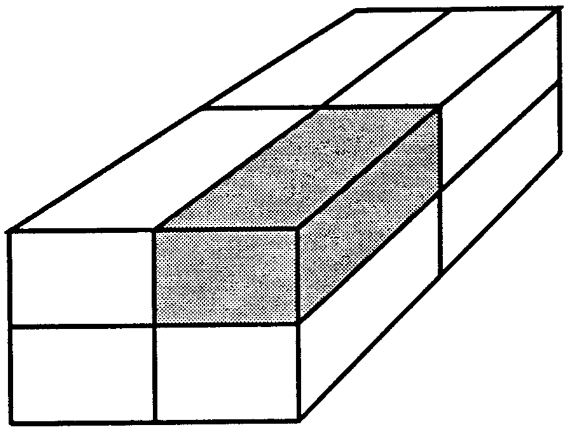

Fig. 4. Schematic illustrating domain decomposition for a problem which will be solved using eight processors and a rectangular three-dimensional mesh. Each processor is assigned a rectangular subgrid.

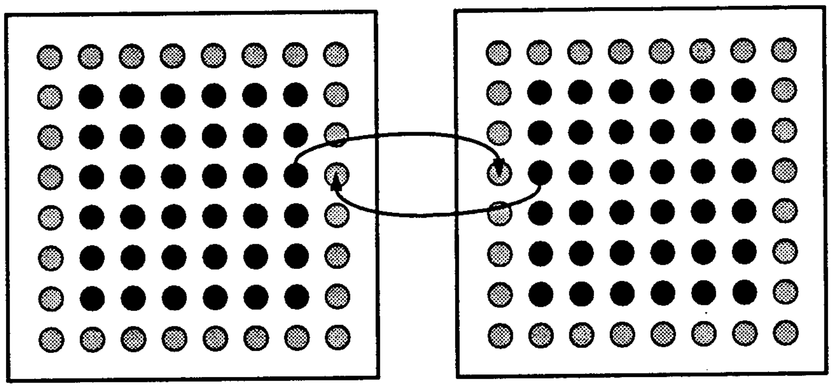

Fig. 5. Interprocessor communication using "ghost" cells. The dark points are grid points on which the solution is calculated by the processor in question, while the lighter points are dummy cells that temporarily hold information communicated by a neighboring processor. Entire sides of the grid are exchanged at one time.

layer of communication cells are added to the boundary of each processor's subgrid (see Fig. 5). These cells are dummy ghost cells that temporarily hold information before it is needed. In a three-dimensional calculation, these ghost cells allow entire faces of the subgrid to be exchanged at one time, thus reducing the number of message start-up delays that are incurred.

A first assessment of the parallel performance of CTH has been completed. This work has shown that the current generation of massively parallel computers can outperform a CRAY Y-MP processor even on problems that are highly vectorized. A two-dimensional version of the hydrodynamics routines that are the basis for CTH has been implemented on both a CM-2 and a nCUBE 2. This version includes the multiple-material mass, momentum and energy equations, the interface tracker and simplified equations of state. Preliminary results are very encouraging (see Table 1). On a 512 by 512 mesh, a 1024 processor nCUBE is five times faster than a CRAY-YMP processor and a 16,384 processor CM-2 is one and a half times faster than the CRAY. The nCUBE can be expanded to 8192 nodes while a CM-2 can have 65,536 processors. It is important to note that 512 by 512 problems are very small for the nCUBE and the CM-2 and that their

TABLE 1. Execution times for a shock physics problem.

Mesh Size	CRAY-YMP	CM-2 16,384 Processors	nCUBE 2 1024 Processors
128 X 128	32 s	32 s	13 s
256 X 256	78 s	78 s	27 s
512 X 512	393 s	249 s	82 s

performance, relative to the CRAY, should improve significantly as bigger problems are solved (large problem sizes on each processor reduce the fraction of time that is used for communication).

The goal of this work is to develop, in the near term, a production level shock physics modeling capability on massively parallel computers. During the next year, detailed equations of state and models for elastic-plastic behavior, explosives and fracture will be added to the two-dimensional, massively parallel version of CTH. Three-dimensional versions of CTH and MESA (which was developed at Los Alamos National Laboratories) will also be implemented on massively parallel machines

STRUCTURAL MECHANICS

A finite-element structural mechanics code, JAC3D, has been implemented on a MIMD machine.[4] The CRAY version of JAC3D code is used extensively within Sandia to analyze nonlinear mechanics problems involving large rotation and strain. Applications include failure of the lower head during hypothetical severe nuclear reactor accidents, integrity of welds on computer chips during manufacturing, the performance of weapons components, and safety analysis of experiments. The model is three-dimensional and quasi-static. Time-dependent loadings can be prescribed and for each load step, a steady solution is calculated. Temperature dependent elastic-plastic and secondary creep behaviors are considered as well moving material interfaces. A non-linear conjugate gradient method is used to solve the discretized equations.

Difficulties encountered in developing a parallel version arose because of the highly irregular geometries of interest. Given a calculational grid, each processor must be assigned a portion of the grid. When dealing with rectangular, cylindrical or spherical grids, mapping to the processors is relatively straightforward. It is much more difficult to map an irregular domain to the processors in a manner that minimizes the number and distance of communications. Three different methods to automatically decompose grids have been investigated:

(1) A recursive-bisection technique. A given grid is divided in half along a line parallel to one of the axes so that equal numbers of grid points are in each subgroup. This process is then repeated on each newly created subgroup. By repeating n times, the calculational grid can be mapped to a hypercube of dimension n.

(2) A graph theory based algorithm. At the start, two opposite extremes of the grid, corresponding to endpoints of a pseudo diameter, are found. Level sets, which are groups of points roughly an equal distance from an extreme point, are determined (here, distance refers to the number of edges that must be transversed to get from one vertex

to another). These sets are then assigned to one of two subgroups, with the selection made according to whichever extreme is the closest. The entire process is then repeated on each newly created subgroup. Once again, the calculational grid is mapped to a hypercube of dimension n by repeating n times.

(3) An iterative technique developed by Kernighan and Lin. At the start of a step, the graph is divided into two equal subsets in an arbitrary manner. Grid points are exchanged between these subsets to minimize interprocessor communication. The process is iterative in nature, and terminates when no exchanges between the subgroups gives a positive gain. Once again, the calculational grid is mapped to a hypercube of dimension n by repeating n times.

A disadvantage of the first method is that it requires the absolute spatial positions of the grid points whereas only connectivity information must be specified to implement the second and third techniques. Each of these schemes assigns roughly the same number of grid points per processor. However, interprocessor communication times for typical problems are very different. The Kernighan and Lin technique gives the minimum communication and the fastest execution time for the application. The recursive-bisection method, on the other hand, results in the slowest problem execution. For a typical structural mechanics problem, there can be as much as a fifty percent difference in problem execution time for the various division methods.

However, the goal is to reduce the total amount of time a researcher must wait for a solution to a problem of interest. This total time is the problem execution time plus the time it takes to map the grid to the processors. In general, more complicated division methods, such as the one devised by Kernighan and Lin, take the longest to execute. These methods are also the ones that result in the fastest problem execution time. Although all divisions are currently done in parallel, the simple recursive bisection technique (method #1 above) is usually the best choice because the division time is a small fraction of the total time. The graph baseds methods offer savings when several calculations are done with the same computational grid (i.e., the division is done only once and the savings in problem execution times accumulate).

Execution times on a CRAY-XMP, a nCUBE 1 and a nCUBE 2 have been compared for a structural mechanics problem involving the manufacturing of a computer chip. Of interest to researchers was the integrity of several welds as the chip was cooled to room temperature. This was a small problem for the nCUBE and the parallel performance, relative to a CRAY, is expected to be higher for larger problems. The problem was solved on the CRAY in about 1700 seconds. Solution times were 2200 seconds on a nCUBE 1 and 350 seconds on a 1024 processor nCUBE 2. Improvements in the compiler and the operating system are expected to lower the nCUBE 2 time below 300 seconds. The parallel version of JAC3D is still in FORTRAN 77 and no attempt was made to optimize any sections of the code using machine language. Bottlenecks did arise on the parallel machines when data was being outputted to a file on the front end machine. However, these problems will be alleviated when the parallel disk system arrives in the near future.

CONCLUSIONS

Massively parallel computers are starting to have a large impact on a wide range of applications. New concepts in heterogenous programming and load balancing for MIMD computers are drastically increasing synthetic aperture radar (SAR) and SDI modeling capabilities. Researchers are also showing that the current generation of massively parallel MIMD and SIMD computers are highly competitive with a CRAY on hydrodynamic and structural mechanics codes that are optimized for vector processors. These results are especially impressive because, in the near future, significant advances in massively parallel systems are anticipated as more processors are added and the computational power of each processor increases.

ACKNOWLEDGEMENTS

The work described in this paper was done by several researchers at Sandia National Laboratories, which is operated for the U.S. Department of Energy under contract number DE-AC04-76DP00789.

REFERENCES

1. J.L. Gustafson, G.R. Montry, and R.E. Benner, "Development of Parallel Methods for a 1024-processor hypercube," **SIAM Journal on Scientific and Statistical Computing, 9**, 609 (1988).

2. J.L. Gustafson, R.E. Benner, M.P. Sears, and T.D. Sullivan, "A Radar Simulation Program for a 1024-Processor Hypercube," in **Proceedings of Supercomputing '89** (1989).

3. R.D. Halbgewachs, J.L. Tomkins, and John VanDyke, "Implementation of Midcourse Tracking and Correlation on Massively Parallel Computers," Report SAND89-2534, Sandia National Laboratories, Albuquerque, NM (1990).

4. C.T. Vaughan, "Implementation of JAC3D on the NCUBE/ten," in **Proceedings of the Fifth Distributed Memory Computer Conference** (1990), pp. 538-544.

5. S.J. Plimpton, "Molecular Dynamics Simulations of Short-Range Force Systems on 1024-Node Hypercubes," in **Proceedings of the Fifth Distributed Memory Computer Conference** (1990), pp. 478-483.

6. D.E. Womble, "A Time-Stepping Algorithm for Parallel Computers," **SIAM Journal on Scientific and Statistical Computing, 11**, 824 (1990).

7. D.E. Womble and B.C. Young, "Multigrid on Massively Parallel Computers," in **Proceedings of the Fifth Distributed Memory Computer Conference** (1990), pp.559-563.

8. J.N. Shadid and R.S. Tuminaro, "Iterative Methods for Nonsymmetric Systems on MIMD Machines," in **Proceedings of the 5th SIAM Conference on Parallel Processing for Scientific Problems,** to appear.

9. J.M. McGlaun, S.L. Thompson, and M.G. Elrick, "A Brief Description of the Three-Dimensional Shock Physics Code CTH," Report SAND89-0607, Sandia National Laboratories, Albuquerque, NM (1989).